Attacking NFC

Charlie Miller
cmiller@openrce.org
0xcharlie

About me

- First to hack the iPhone and G1 Android phone
- Winner of CanSecWest Pwn2Own: 2008-2011
- Author
  - Fuzzing for Software Security Testing and Quality Assurance
  - The Mac Hacker's Handbook
  - The iOS Hacker's Handbook
- PhD, CISSP, GCFA, etc.
- Not a member of iOS Developer Progra

Agenda

- Motivations
- NFC basics
- Low level attacks
- High level attacks
- Potential attacks and demos

Motivation

Samsung brings NFC tagging to the Galaxy S3

Launches programmable Tectile stickers

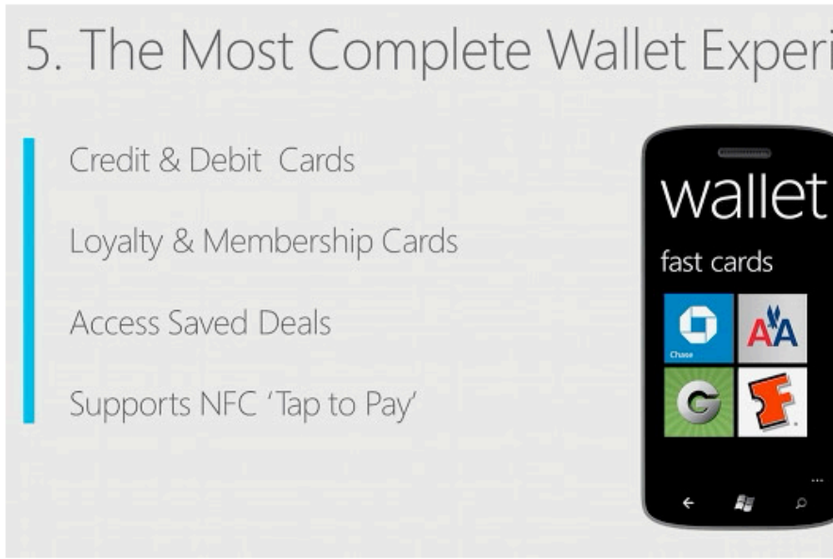
By **Carly Page**

NFC to come as standard on all Nokia phones

Future handsets to have the chip inside

Microsoft brings NFC payments and loyalty c
Windows Phone 8

By Donald Melanson posted Jun 20th 2012 12:35PM



Apple patent suggests NFC for iPhone 5

NFC Predictions by Deloitte : 200 Million NFC Devices by 2012 End

- NFC is coming to a phone near you
- NFC represents new "server-side" attack surface

Other NFC work

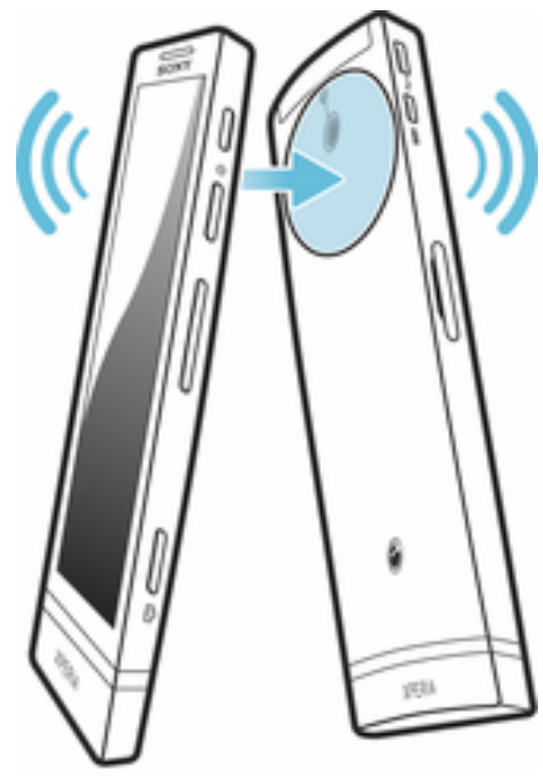
- zvelo: Google wallet PIN brute forcing
- Intrepidus Group: Misdirecting four square, malware and NFC intents, parking meters, subway passes etc
- Ruhr University: MIFARE encryption cracking
- MWR: bus passes, gym memberships, etc
- Collin Mulliner: URL spoofing, snacks from vending machines

NFC basics

- Set of communication protocols based on RFID standards including ISO 14443
- 13.56 Mhz operating frequency +/- 7kHz
- Operating range less than 4 cm
- Data rates: 106, 212, 424 kbits/s

Communication modes

- Passive
  - Initiator provides carrier fields
    - Target modulates existing field
- Active (P2P)
  - Initiator and target



How close

- Close but not touching
- Can read card through wallet in pocket

NFC and the screen

- NFC is typically on when the phone's screen is on
  - i.e. not when phone is "asleep"
- ICS - only on when phone is unlocked
- Can wake up the phone if you know the target's phone number

NFC attack vectors

- “The subway attack”
- “Card skimming”

Subway attack

When a subway isn't handy

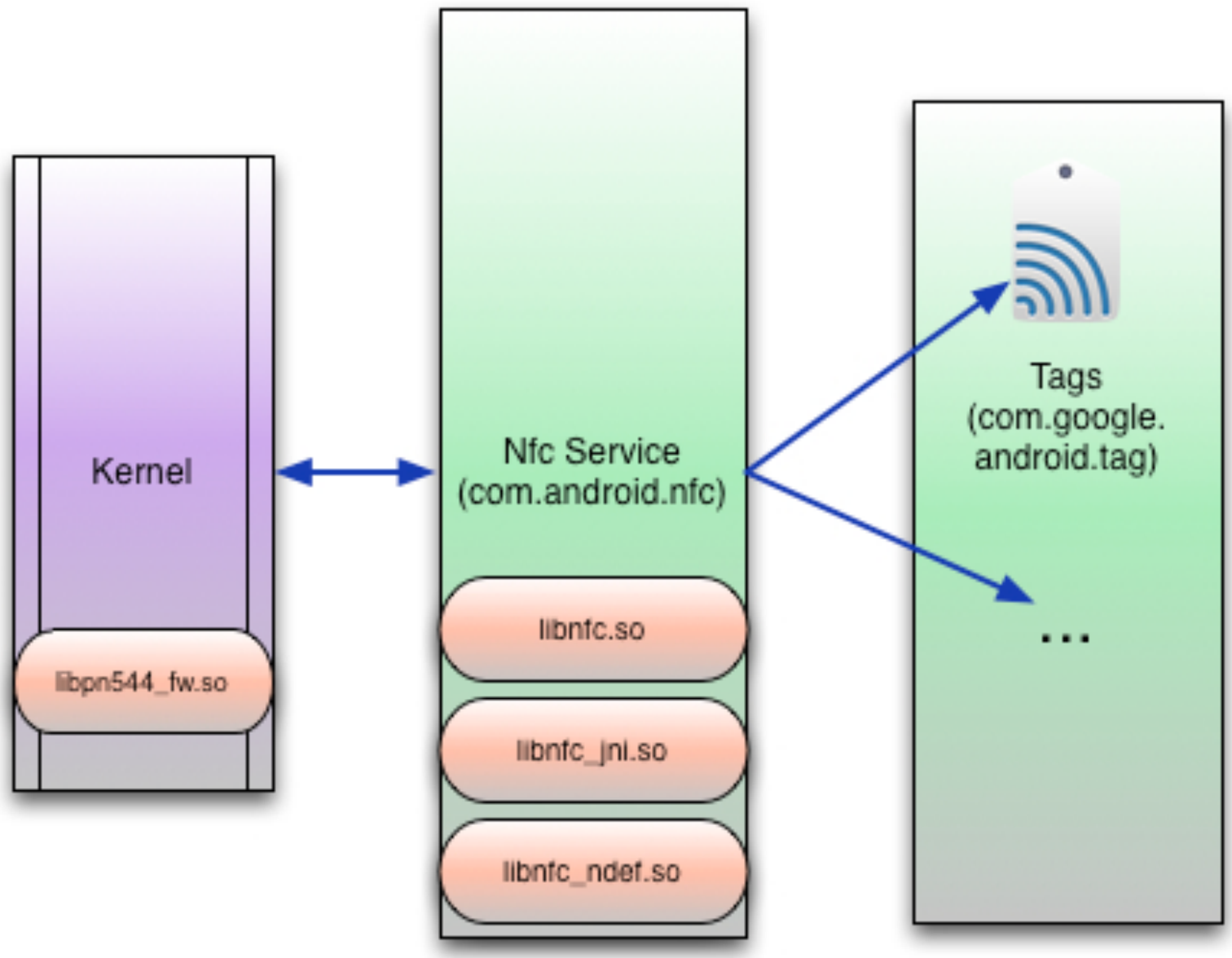
ATM card skimmers

NFC “card skimming”

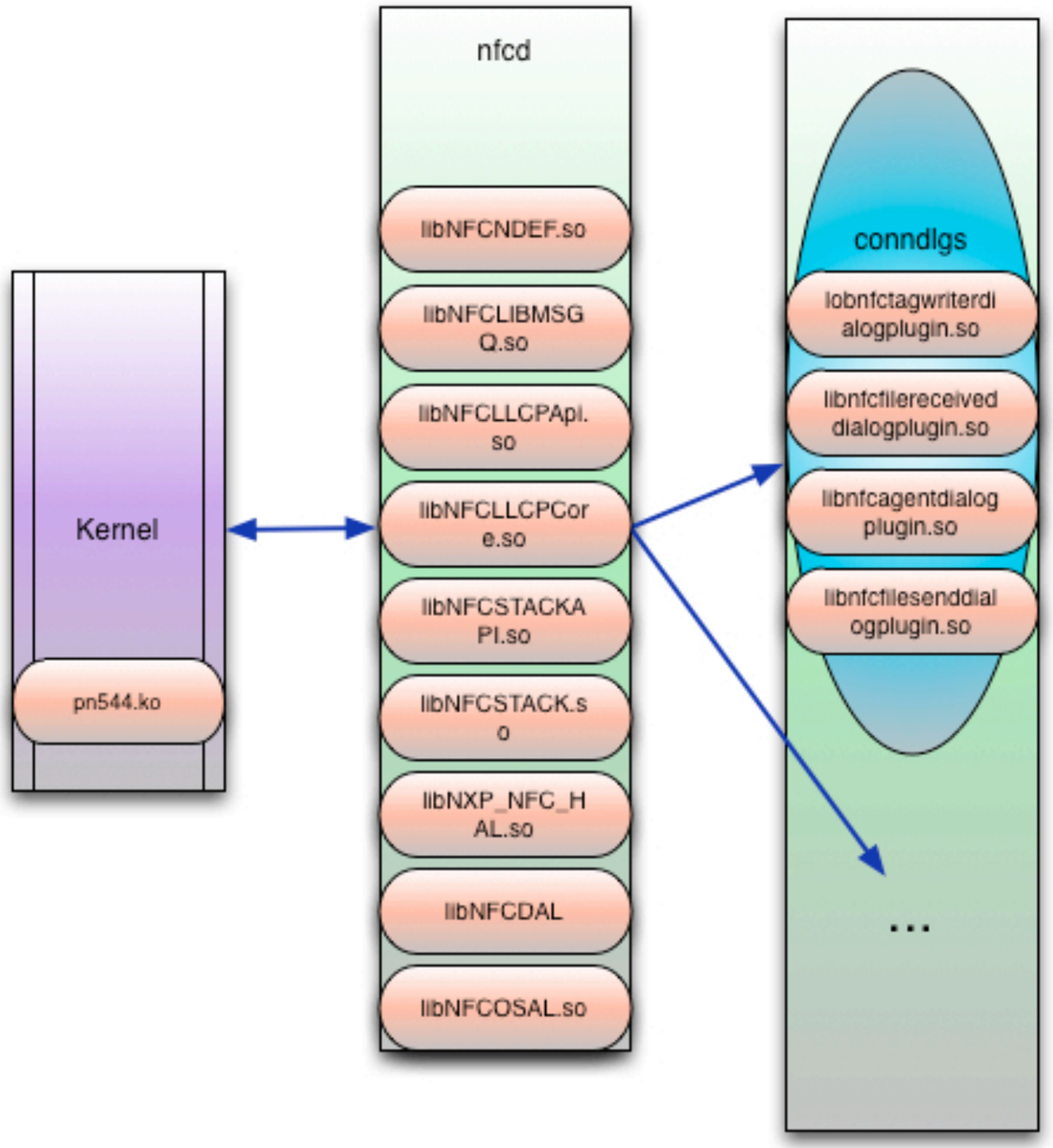
Where might the bugs be?

- Low level
  - The actual NFC parsing code in the (firmware), driver, NFC service, etc
- Higher level
  - Applications which consume data (without user interaction)

Android NFC stack



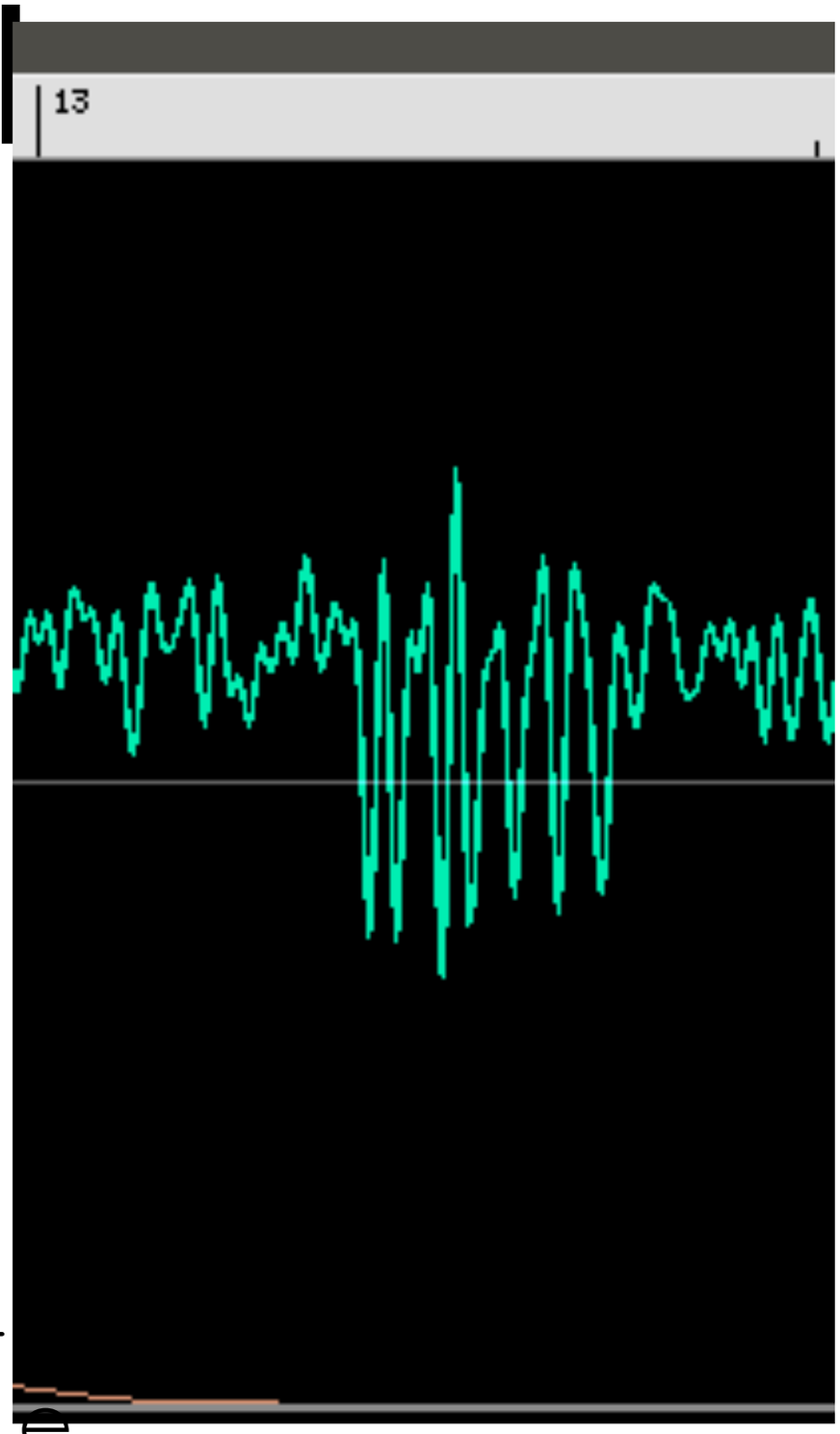
MeeGo NFC stack



Specs


Application Layer
NFC Forum NDEF Messages
Tag Types
Type 1 Tag
NFC Forum Formatted Tag
Type 2 Tag
Type 4 Tag
Type 3 Tag
Examples of products
Topaz (Innovision)
MIFARE Std 1K/4K (NXP)
MIFARE Ultralight (NXP)
MIFARE DESFire (NXP)
16RF (STM)
SLE66CL (Infineon)
FeliCa non-secure (Sony)
NFC Device
Applicative Protocol
ISO/IEC 7816-4 (APDUs)
Protocol
Topaz protocol
MIFARE protocol
MIFARE Ultralight protocol
ISO 14443 A-4
ISO 14443 B-4
LLCP (Peer 2 Peer)
Initialization Anticollision Protocol Activation
No Anti-collision
ISO 14443 A-3 / ISO18092
ISO 14443 B-3
FeliCa / JIS X 6319-4 and ISO 18092
RF
ISO 14443 A-2 / ISO18092
ISO 14443 B-2
ISO 18092
Physical Characteristics
ISO 14443 A-1 / ISO18092
ISO 14443 B-1
NFC A
NFC B
NFC F
P2P
NFC Forum specification
Vendor specific
International standard

Physical



- 100% ASK using Manchester decoding
  - 0x26 = SENS_REQ (ISO 14443-3)

```
...1111101011010110110111
          s 0 1 1 0 0 1 0 e
0100110 = 0x26 = SENS_REQ (ISO 14443-3)
```

Protocol layer

- Type 1 (Topaz)
- MIFARE Classic
- Type 2 (MIFARE Ultra
- Type 3
- Type 4 (DESFire)
- LLCP (P2P)

Type 2: MIFARE UL

- Command set: **READ**, WRITE, SECTOR

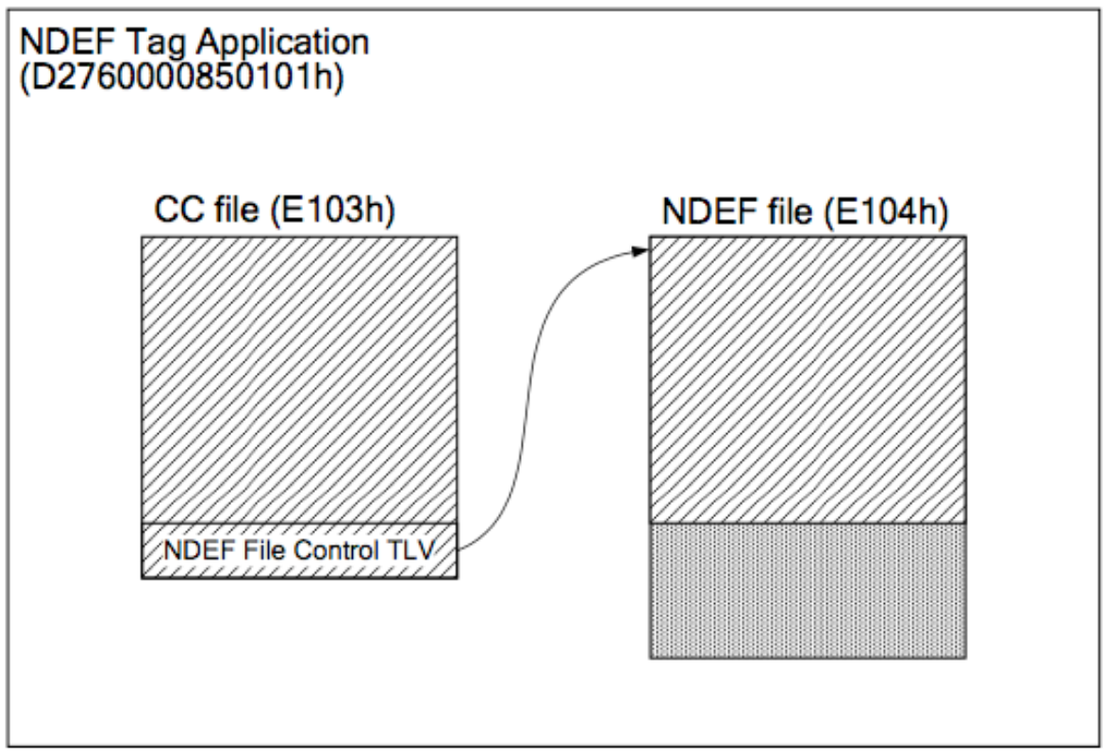
Byte Number	0	1	2	3	Block
UID / Internal	Internal0	Internal1	Internal2	Internal3	0
Serial Number	Internal4	Internal5	Internal6	Internal7	1
Internal / Lock	Internal8	Internal9	Lock0	Lock1	2
CC	CC0	CC1	CC2	CC3	3
Data	Data0	Data1	Data2	Data3	4
Data	Data4	Data5	Data6	Data7	5
Data	Data8	Data9	Data10	Data11	6
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	.
Data	..	..	..	..	n
Lock / Reserved	..	..	..	..	.
Lock / Reserved	..		..	..	.
Lock / Reserved	..	..	..	..	k

Capability container

NDEF data

Type 4: DESFire

- Command set: **SELECT**, **READ**, UPDATE



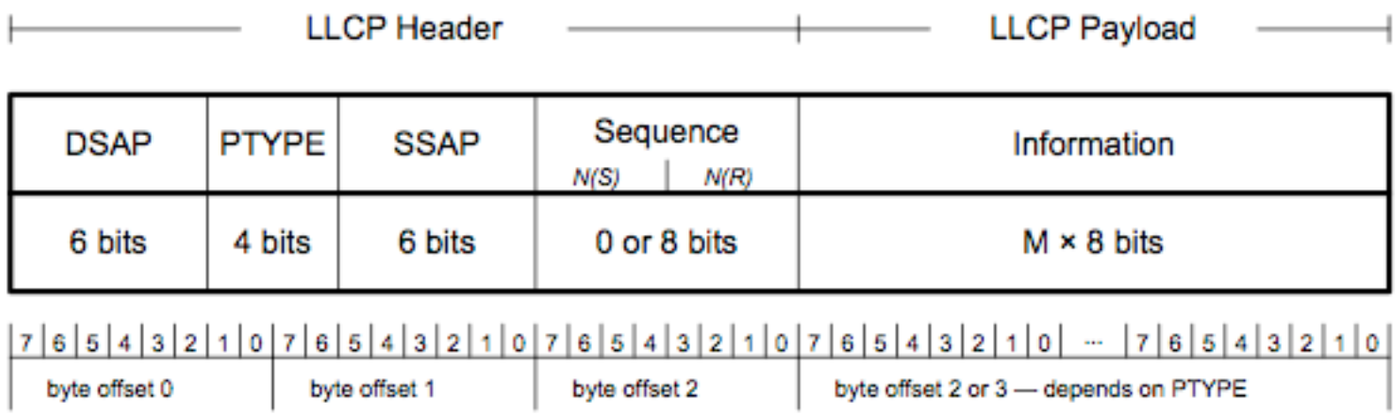
Application

Empty Memory Area

Used Memory Area

LLCP

- PDU types
  - **SYMM**, PAX, AGF, UI, **CONNECT**, DISC,



DSAP = Destination service access point address field

PTYPE = Payload data unit (PDU) type field

SSAP = Source service access point address field

Sequence = Sequence field (8 bits for formats that include sequence numbers, and 0 bits for formats that do not)

Information = Information field (M is an integer value between and including 0 and the maximum information unit MIU defined in this specification; × denotes multiplication)

Application layer

- NFC Data Exchange Format (NDEF)
- Binary message format
- Different identifiers to describe types such as URI's, MIME types, NFC-specific type
- Specification for NDEF and each well known type

```
SENS_REQ
     26
SENS_RES (NFCID1 size: double (7 bytes), Bit frame SDD)
     44  00

SDD_REQ (CL1, SEL_PAR bc=2)
     93  20
SDD_RES (CT, 04-e3-ef, BCC)
     88  04  e3  ef  <80>

SEL_REQ (CL1, NFCID1)
     93  70  88  04  e3  ef  80  <99  73>
SEL_RES - NFCID not complete, type 2 tag
     04  <da  17>

SDD_REQ (CL2, SEL_PAR bc=2)
     95  20
SDD_RES (a2-ef-20-80 BCC)
     a2  ef  20  80  <ed>

SEL_REQ (CL2, NFCID2)
     95  70  a2  ef  20  80  ed  <72  c8>
SEL_RES - NFCID complete, type 2 tag
     00  <fe  51>
-------------------------------------------------------------------------
READ - 08
     30  08  <4a  24>
READ Response
     74  72  61  6c  69  67  68  74  3f  fe  00  00  e4  f2  e3  01  <06  d5>
READ - 03
     30  03  <99  9a>
READ Response
     e1  10  06  00  03  17  d1  01  13  54  02  65  6e  73  75  70  <b1  62>
READ - 04
     30  04  <26  ee>
READ Response
     03  17  d1  01  13  54  02  65  6e  73  75  70  2c  20  75  6c  <2a  00>
READ - 05
     30  05  <af  ff>
...
```

serial number, memory protections, CC, etc

NDEF data

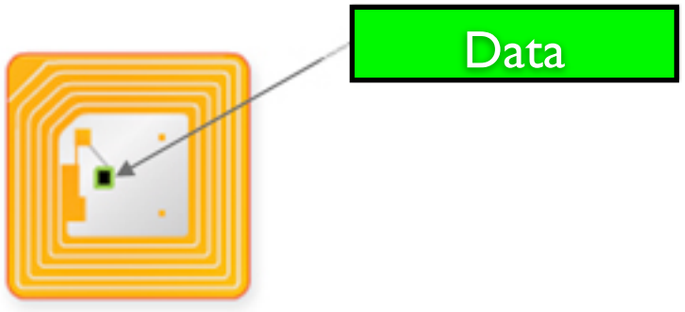
Extracted NDEF data

```
03  17  d1  01  13  54  02  65  6e  73  75  70  2c  20  75  6c  74  72  61
6c  69  67  68  74  3f  fe

03 NDEF Message
17 length
    Record 1:
        d1 - MB, ME, SR, TNF=”NFC Forum well-known type”
        01 Type length
        13 Payload length
            54 Type - “T”
                02 - Status byte - Length of IANA lang code
                65 6e - language code = “en”
                73  75  70  2c  20  75  6c  74  72  61  6c  69  67  68  74  3f
= “sup, ultralight?” - text
    Record 2:
        fe Terminator NDEF
```

Fuzzing NFC

Delivering test cases



Test case generation

Monitoring device

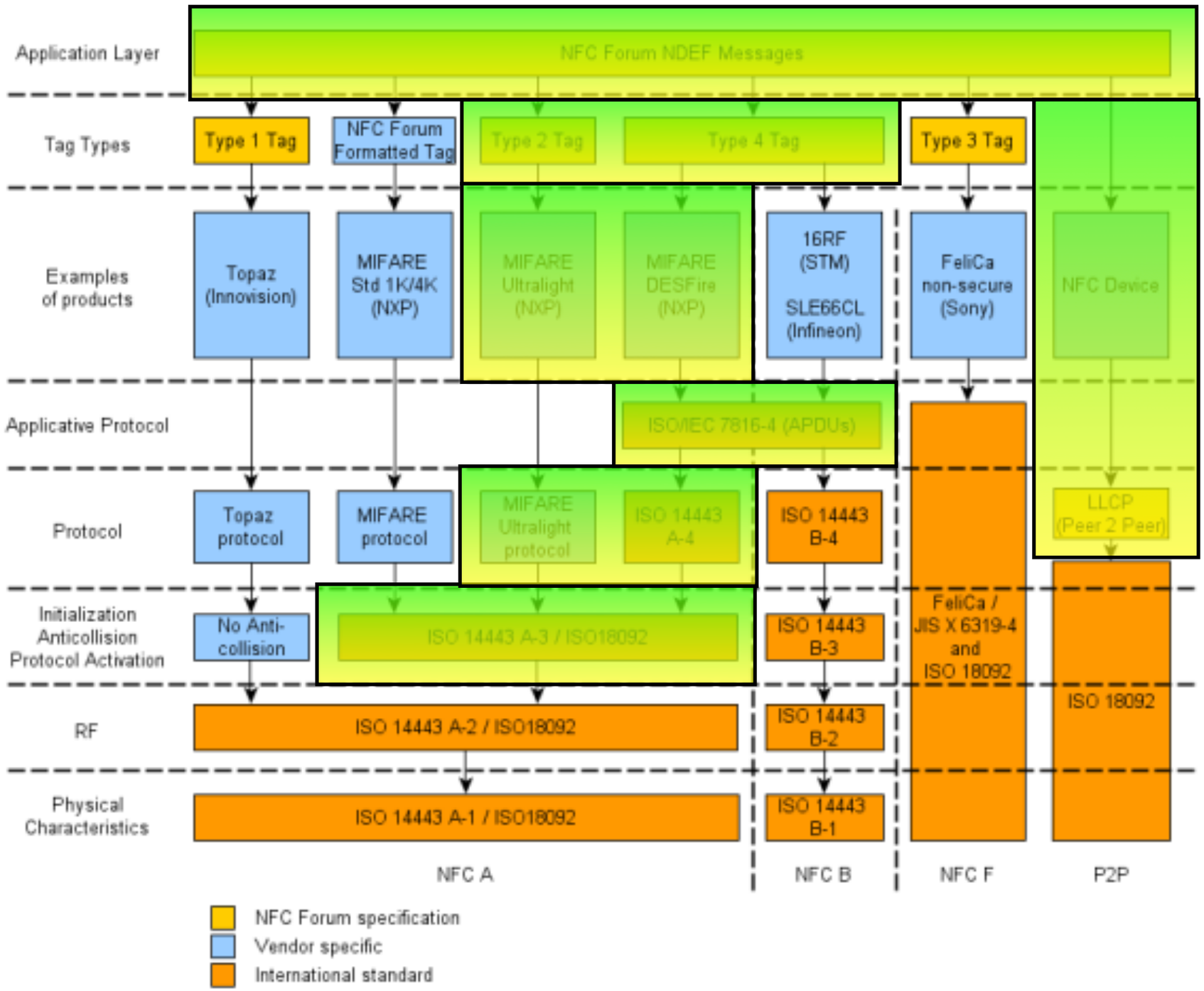
NFC readers that work

libnfc.org

- ACS ACR122U with libnfc 1.5.1
  - Card emulation for type 2 and type 4 cards
- SCL3711 with nfcpy
  - LLCP with either SNEP or NPP

Python module for near field communication

Fuzzable with this setup



Fuzzing in action

Low level fuzzing performed

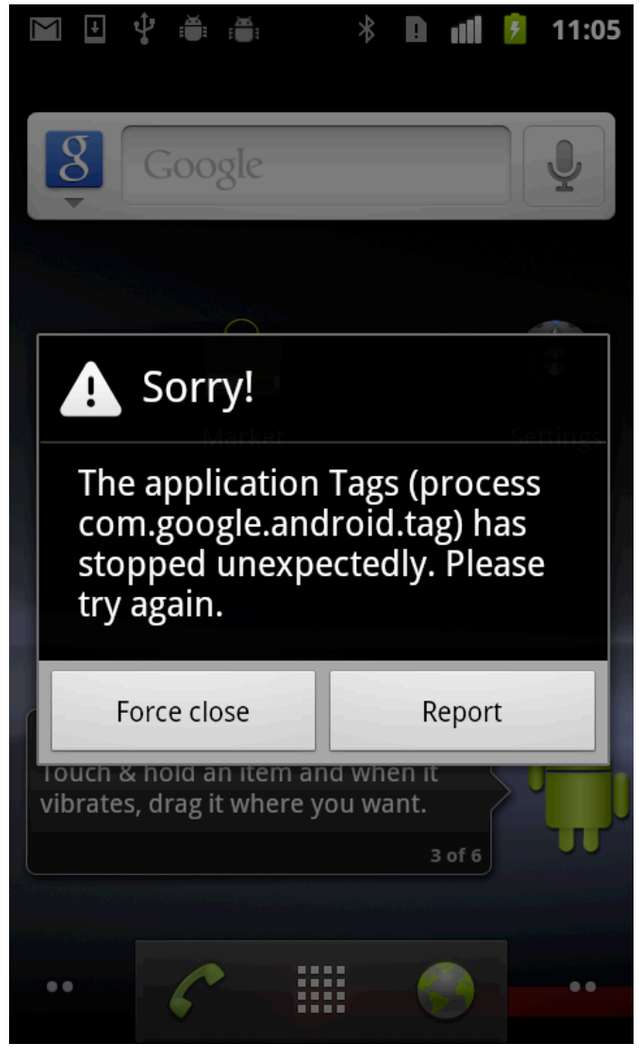
- Fuzz targets
  - Nexus S running Android 2.3.3 Gingerbread
  - Nokia N9 1.2 Harmattan PR 1.2
- 30,000 - 60,000 test cases
- Each test case took between 5-10 (or more)

Test cases used

	Android test cases	Meego test cases
Type 2 - low level	4000	4000
Type 4 - low level	4000	4000
LLCP - Connect	2000	2000
LLCP - I	2000	2000
NDEF -bitflip	9000	9000
NDEF - short text	1626	1626
NDEF - short URI	538	538
NDEF - short SMS	1265	1265
NDEF - short SP	2440	2440
NDEF - short BT	1246	1246
NDEF - long text	2440	2440
NDEF - long vcard	32572	15062
Total	52362	34852

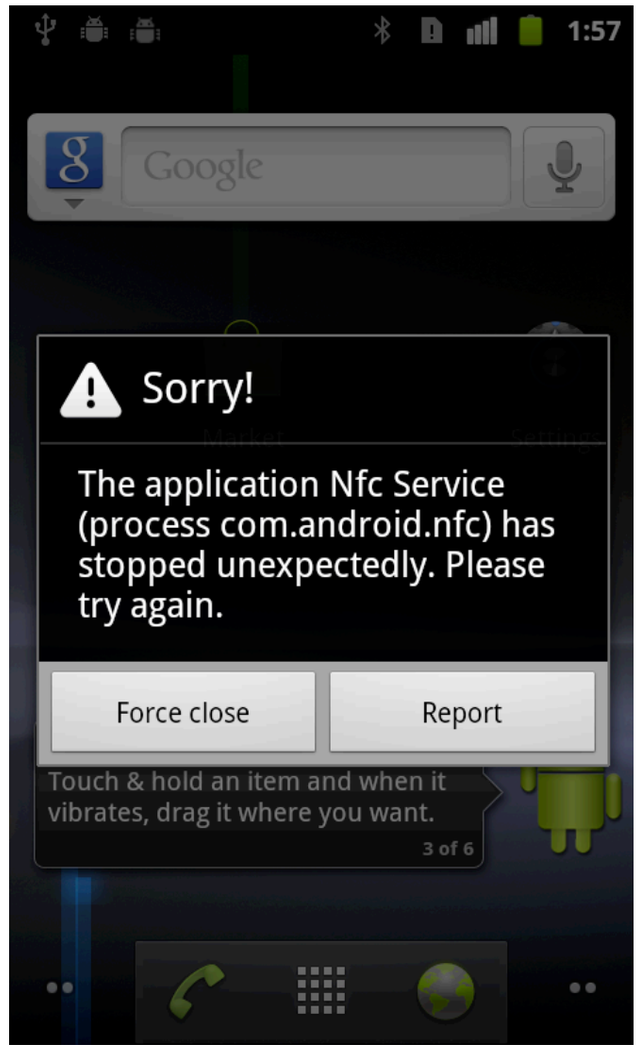
Android Java Exceptions

```
E/NfcService(17875): failed to parse record
E/NfcService(17875): java.lang.ArrayIndexOutOfBoundsException
E/NfcService(17875):  at com.android.nfc.NfcService
$NfcServiceHandler.parseWellKnownUriRecord(NfcService.java:2570)
E/NfcService(17875):  at com.android.nfc.NfcService
$NfcServiceHandler.setTypeOrDataFromNdef(NfcService.java:2616)
E/NfcService(17875):  at com.android.nfc.NfcService
$NfcServiceHandler.dispatchTagInternal(NfcService.java:2713)
```



More Android Java Exceptions

```
D/NdefPushServer( 3130): java.io.IOException
D/NdefPushServer( 3130):  at
com.android.internal.nfc.LlcpSocket.receive(LlcpSocket.java:193)
D/NdefPushServer( 3130):  at
com.android.nfc.ndefpush.NdefPushServer
$ConnectionThread.run(NdefPushServer.java:70)
D/NdefPushServer( 3130): about to close
W/dalvikvm( 3130): threadid=8: thread exiting with uncaught
exception (group=0x40015560)
E/AndroidRuntime( 3130): FATAL EXCEPTION: NdefPushServer
E/AndroidRuntime( 3130): java.lang.NegativeArraySizeException
E/AndroidRuntime( 3130):  at
com.android.nfc.ndefpush.NdefPushProtocol.<init>(NdefPushProtocol
.java:97)
E/AndroidRuntime( 3130):  at
com.android.nfc.ndefpush.NdefPushServer
$ConnectionThread.run(NdefPushServer.java:86)
```



Android null ptr deref

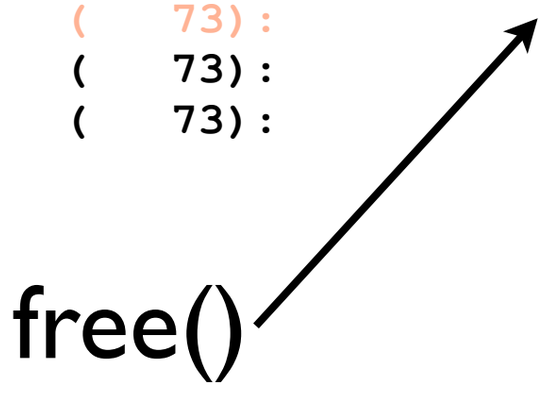
- Send a CC PDU without first establishing a

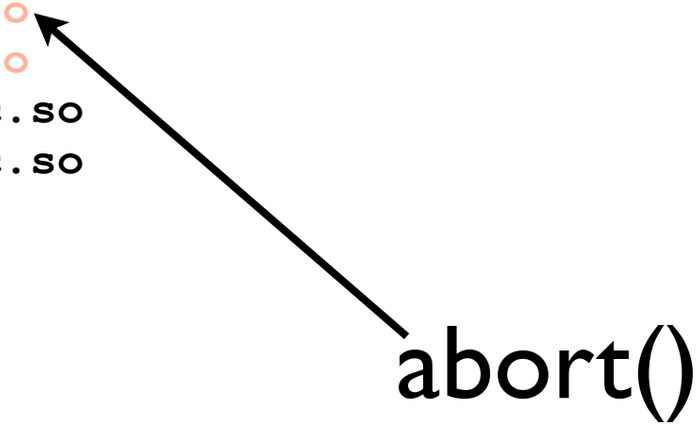
```
BAD PDU: 05a0060f636f6d2e616e64726f69642e6e7070
```

```
0x80528f1c in Handle_ConnectionOriented_IncommingFrame ()
   from /home/cmiller/debugging/libnfc.so
 ...
(gdb) x/i $pc
0x80528f1c <Handle_ConnectionOriented_IncommingFrame+952>:     stmia   r3, {r0, r1}
(gdb) print /x $r3
$3 = 0x0
```

Android Double Free

```
D/NdefPushServer(13178): created LLCP service socket
D/NdefPushServer(13178): about to accept
D/NFC JNI (13178): Discovered P2P Target
D/NfcService(13178): LLCP Activation message
E/NFC JNI (13178): phLibNfc_Llcp_CheckLlcp() returned 0x00ff[NFCSTATUS_FAILED]
I/DEBUG   (   73): *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***
I/DEBUG   (   73): Build fingerprint: 'google/sojua/crespo:2.3.3/GRI54/105536:user/
release-keys'
I/DEBUG   (   73): pid: 13178, tid: 13178  >>> com.android.nfc <<<
I/DEBUG   (   73): signal 11 (SIGSEGV), code 1 (SEGV_MAPERR), fault addr 0000000c
I/DEBUG   (   73):  r0 afd46494  r1 00000004  r2 00000000  r3 afd46450
I/DEBUG   (   73):  r4 00295530  r5 afd46450  r6 00000000  r7 40002410
I/DEBUG   (   73):  r8 00000001  r9 0000008a  10 00000002  fp bed9725c
I/DEBUG   (   73):  ip afd46474  sp bed97220  lr afd10e60  pc afd13d06  cpsr 00000030
...
I/DEBUG   (   73):          #00  pc 00013d06  /system/lib/libc.so
I/DEBUG   (   73):          #01  pc 000144be  /system/lib/libc.so
I/DEBUG   (   73):          #02  pc 0004375c  /system/lib/libnfc.so
I/DEBUG   (   73):          #03  pc 00042b84  /system/lib/libnfc.so
```





Source code

```
2047 /* Llcp methods */
2048
2049 static jboolean com_android_nfc_NfcManager_doCheckLlcp(JNIEnv *e, jobject o)
2050 {
2051    NFCSTATUS ret;
2052    jboolean result = JNI_FALSE;
2053    struct nfc_jni_native_data *nat;
2054    struct nfc_jni_callback_data  *cb_data;
2055
2056
2057    CONCURRENCY_LOCK();
2058
2059    /* Memory allocation for cb_data */
2060    cb_data = (struct nfc_jni_callback_data*) malloc (sizeof(nfc_jni_callback_data));
...
2081    if(ret != NFCSTATUS_PENDING && ret != NFCSTATUS_SUCCESS)
2082    {
2083       LOGE("phLibNfc_Llcp_CheckLlcp() returned 0x
%04x[%s]", ret, nfc_jni_get_status_name(ret));
2084       free(cb_data);
2085       goto clean_and_return;
2086    }
...
2101 clean_and_return:
2102    nfc_cb_data_deinit(cb_data);
2103    CONCURRENCY_UNLOCK();
2104    return result;
2105 }
```

Status of vulnerability

- Fixed in ICS (4.0.1) by Google (independent of me)
- Gingerbread devices are still vulnerable
  - 92% of currently deployed Android devices

Other crashes

```
I/DEBUG   (   73):          #00  pc 00015ca4  /system/lib/libc.so <libc_android_abort>
I/DEBUG   (   73):          #01  pc 00013e08  /system/lib/libc.so <dlmalloc>
I/DEBUG   (   73):          #02  pc 0001423e  /system/lib/libc.so <???>
I/DEBUG   (   73):          #03  pc 000142ac  /system/lib/libc.so <dlrealloc>
I/DEBUG   (   73):          #04  pc 0001451a  /system/lib/libc.so <realloc>
I/DEBUG   (   73):          #05  pc 0001abf0  /system/lib/libbinder.so
<android::Parcel::continueWrite>
I/DEBUG   (   73):          #06  pc 0001ad0c  /system/lib/libbinder.so
<android::Parcel::growData>
I/DEBUG   (   73):          #07  pc 0001ae68  /system/lib/libbinder.so
<android::Parcel::writeInplace>
DEBUG   (   73):          #08  pc 0001aea8  /system/lib/libbinder.so
<android::Parcel::writeString16>
DEBUG   (   73):          #09  pc 0001aed4  /system/lib/libbinder.so
<android::Parcel::writeString16>
DEBUG   (   73):          #10  pc 0001aef8  /system/lib/libbinder.so
<android::Parcel::writeInterfaceToken>
```

Other crashes

```
I/DEBUG   (   73):          #00  pc 00015ca4  /system/lib/libc.so   <libc_android_abort>
I/DEBUG   (   73):          #01  pc 00013614  /system/lib/libc.so   <dlfree>
I/DEBUG   (   73):          #02  pc 000144da  /system/lib/libc.so   <free>
I/DEBUG   (   73):          #03  pc 0004996e  /system/lib/libdvm.so <dvmDestroyJNI>
I/DEBUG   (   73):          #04  pc 00053fda  /system/lib/libdvm.so
<dvmDetachCurrentThread>
I/DEBUG   (   73):          #05  pc 000494da  /system/lib/libdvm.so <???>
I/DEBUG   (   73):          #06  pc 00005310  /system/lib/libnfc_jni.so
<nfc_jni_client_thread>
I/DEBUG   (   73):          #07  pc 000118e4  /system/lib/libc.so   <_thread_entry>
```

Other crashes

```
I/DEBUG   (   73):          #00  pc 00013256  /system/lib/libc.so   <dlfree>
I/DEBUG   (   73):          #01  pc 000144da  /system/lib/libc.so   <free>
I/DEBUG   (   73):          #03  pc 0004996e  /system/lib/libdvm.so <dvmDestroyJNI>
I/DEBUG   (   73):          #04  pc 00053fda  /system/lib/libdvm.so
<dvmDetachCurrentThread>
I/DEBUG   (   73):          #05  pc 000494da  /system/lib/libdvm.so <???>
I/DEBUG   (   73):          #06  pc 00005310  /system/lib/libnfc_jni.so
<nfc_jni_client_thread>
I/DEBUG   (   73):          #07  pc 000118e4  /system/lib/libc.so   <_thread_entry>
```

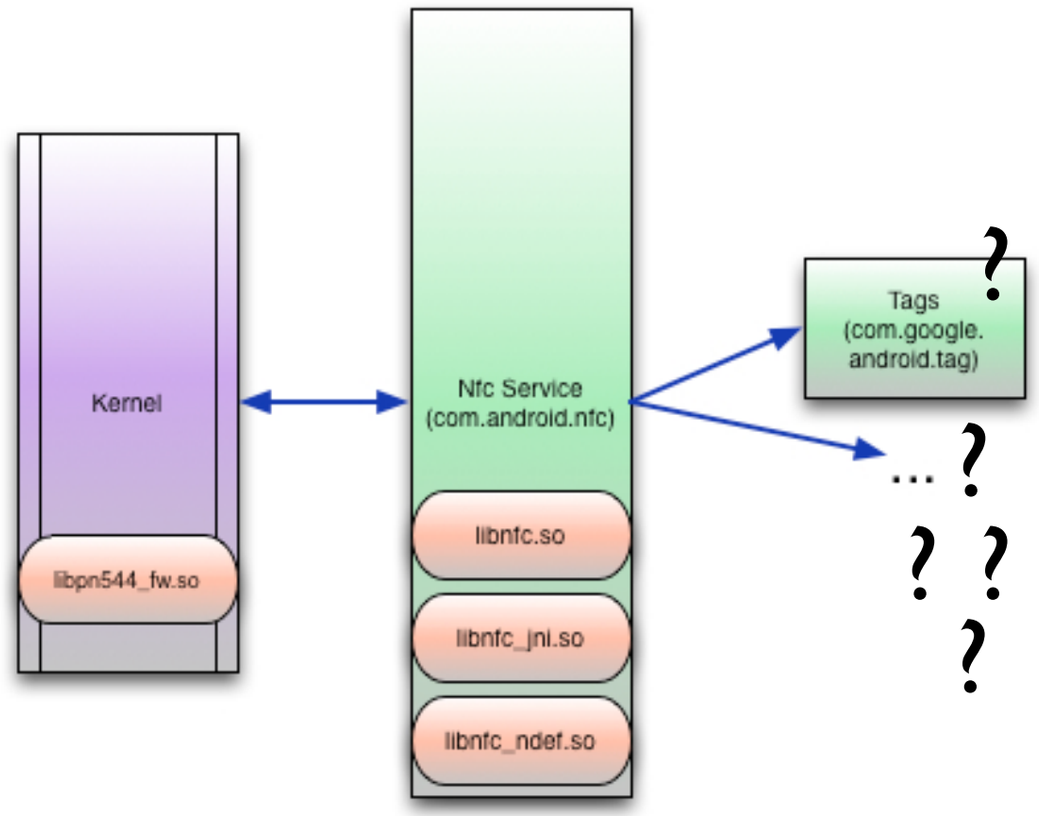
crash occurs in unlink_large_chunk in dlfree()
when invalid “back” ptr is referenced

Other crashes

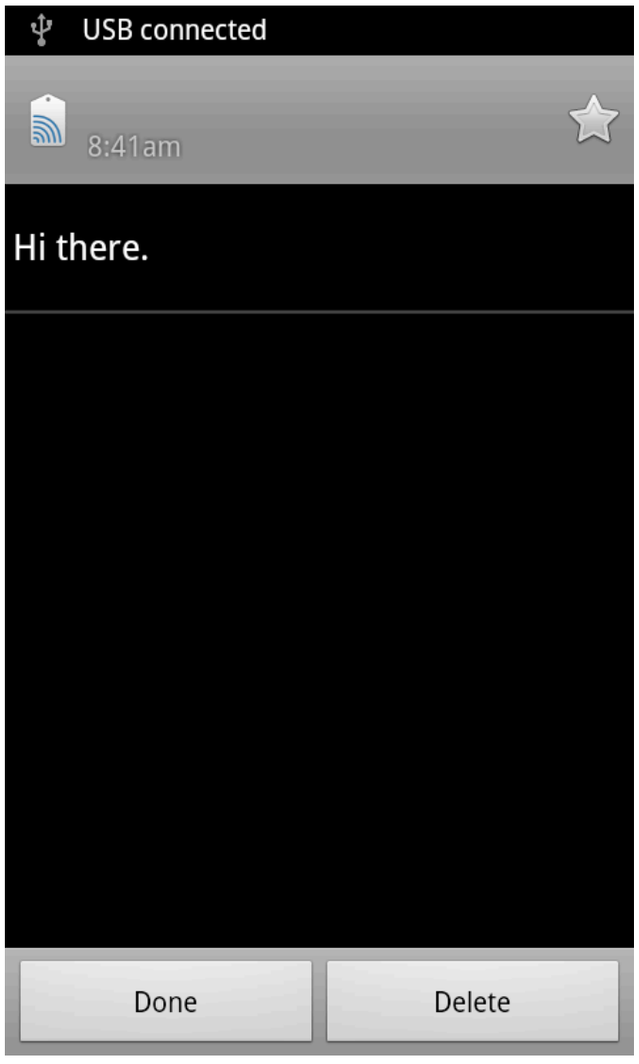
```
I/DEBUG   (   73):          #00  pc 00015ca4  /system/lib/libc.so    <libc_android_abort>
I/DEBUG   (   73):          #01  pc 00013e08  /system/lib/libc.so    <dlmalloc>
I/DEBUG   (   73):          #02  pc 000144be  /system/lib/libc.so    <calloc>
I/DEBUG   (   73):          #03  pc 000509c8  /system/lib/libdvm.so
<dvmInitReferenceTable>
I/DEBUG   (   73):          #04  pc 000533f8  /system/lib/libdvm.so  <???>
I/DEBUG   (   73):          #05  pc 00053454  /system/lib/libdvm.so
<dvmAttachCurrentThread>
```

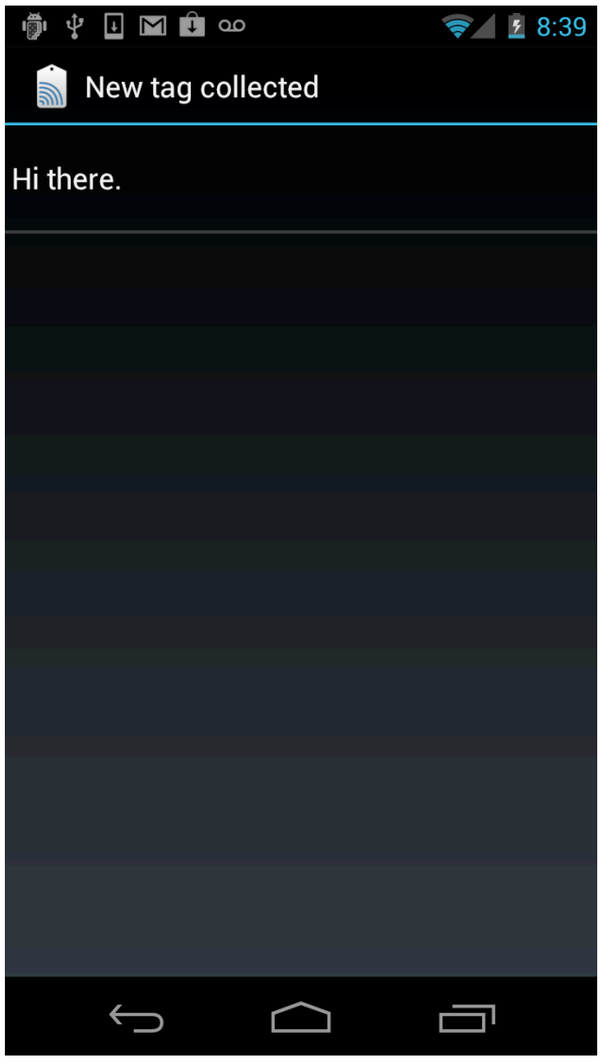
Beyond the NFC stack

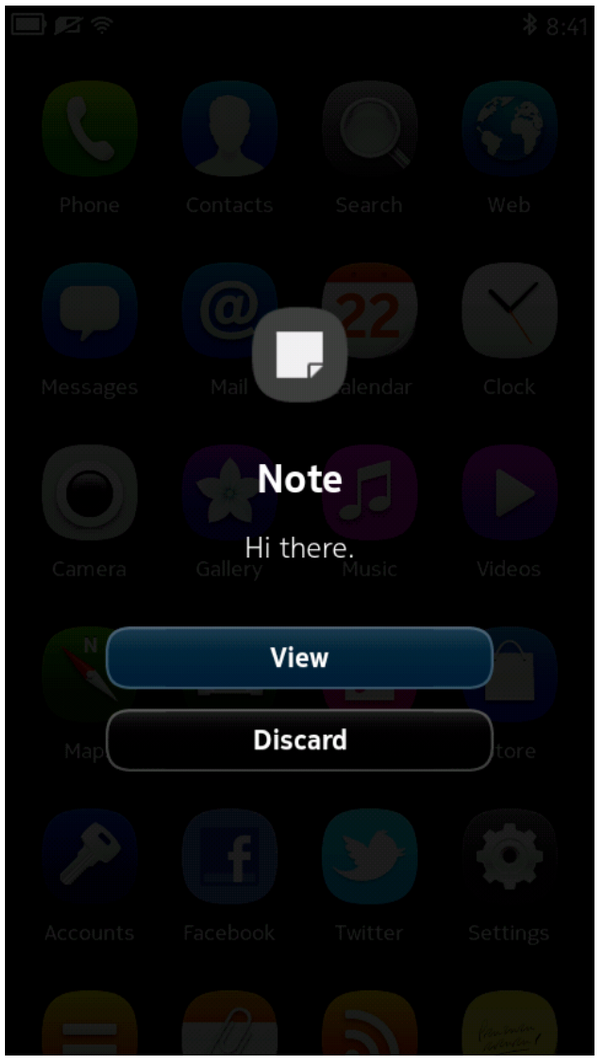
- What applications handle the actual NFC data
  - by default
  - without user interaction



At first glance - boring!

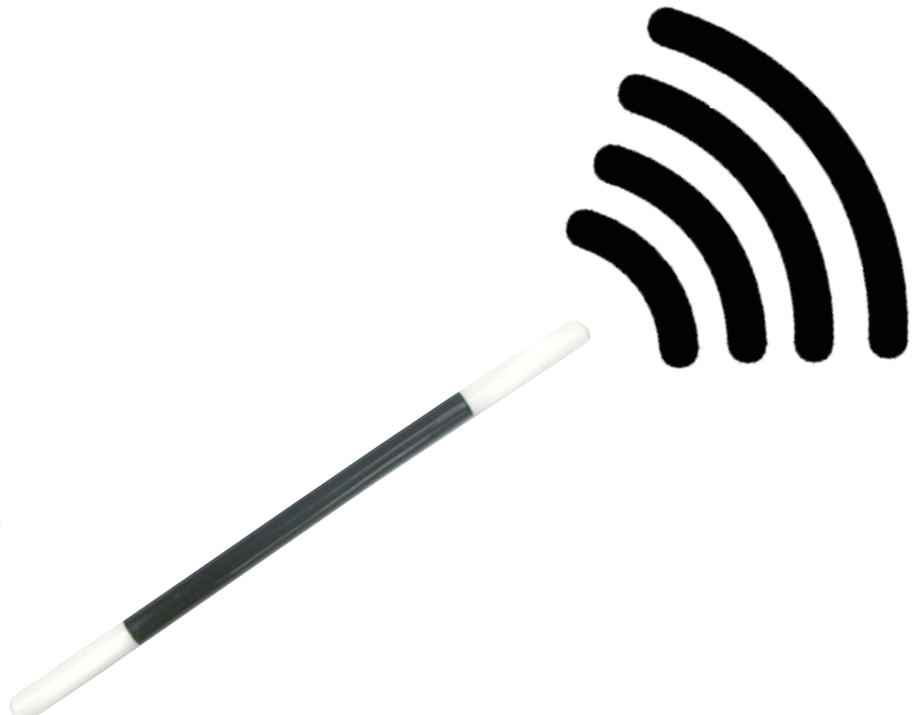






Cool NFC magic

- Android: Beam
- Nokia: Content sharing
- Nokia: Bluetooth pairing
- Samsung S-Beam

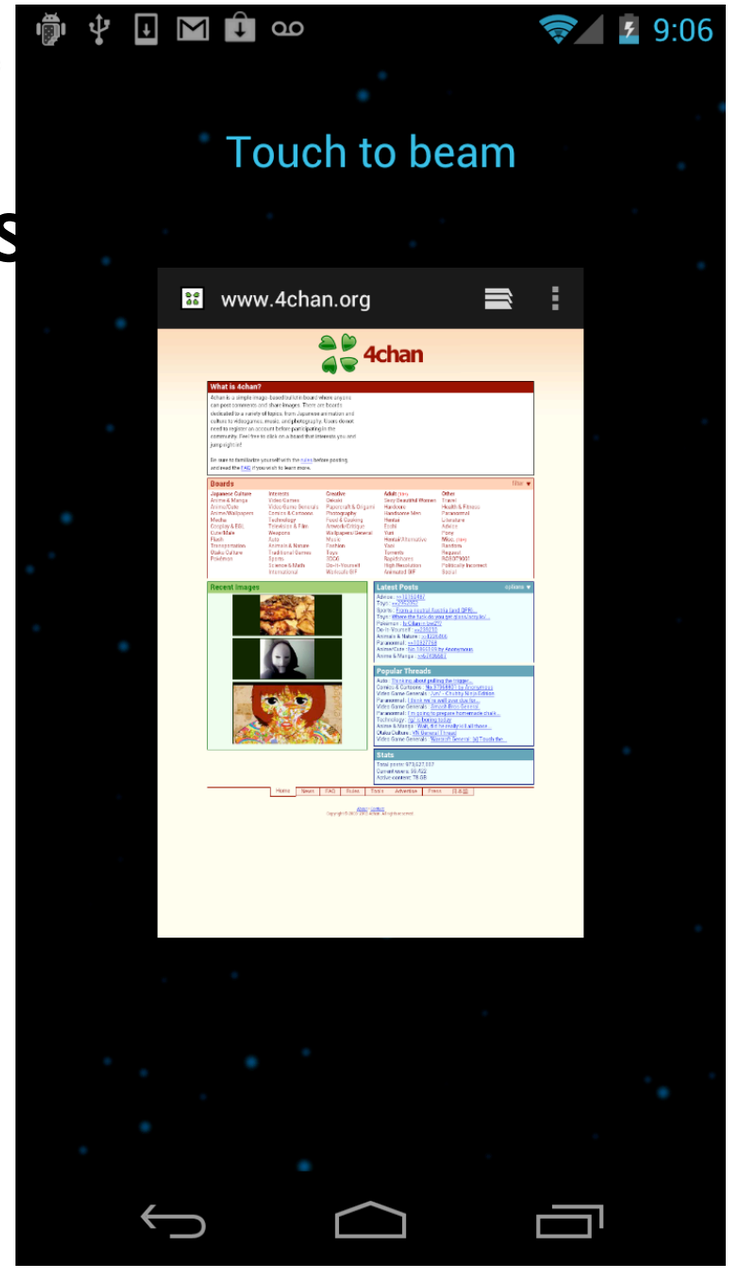


Android Beam

- Introduced in ICS
- Two devices can share content via NFC
- LLCP + SNEP, fallback to LLCP + NPP

More Android Beam

- Implemented with Android intents
  - Browser, Contacts, and Tags regis




```
<!-- Accept inbound NFC URLs at a low priority -->
<intent-filter android:priority="-101">
    <action
android:name="android.nfc.action.NDEF_DISCOVERED" />
    <category
android:name="android.intent.category.DEFAULT" />
    <data android:scheme="http" />
    <data android:scheme="https" />
</intent-filter>
```

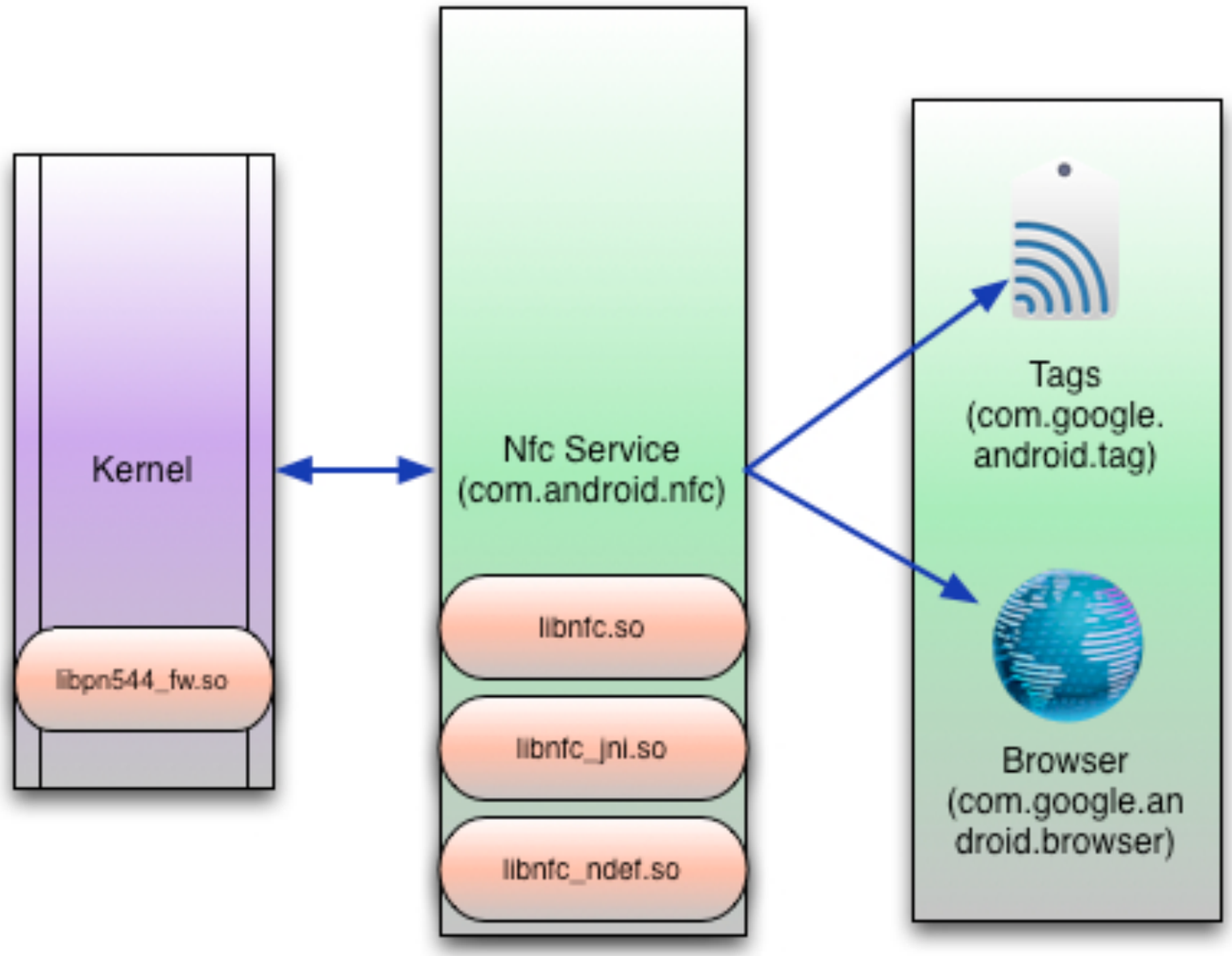
Bigger attack surface

Type	File format
Web related	html
	css
	js
	xml
Image	bmp
	gif
	ico
	jpg
	wbmp
	svg
	png
Audio	mp3
	aac
	amr
	ogg
	wav
Video	mp4
	3pg
Font	ttf
	eot

Demo!

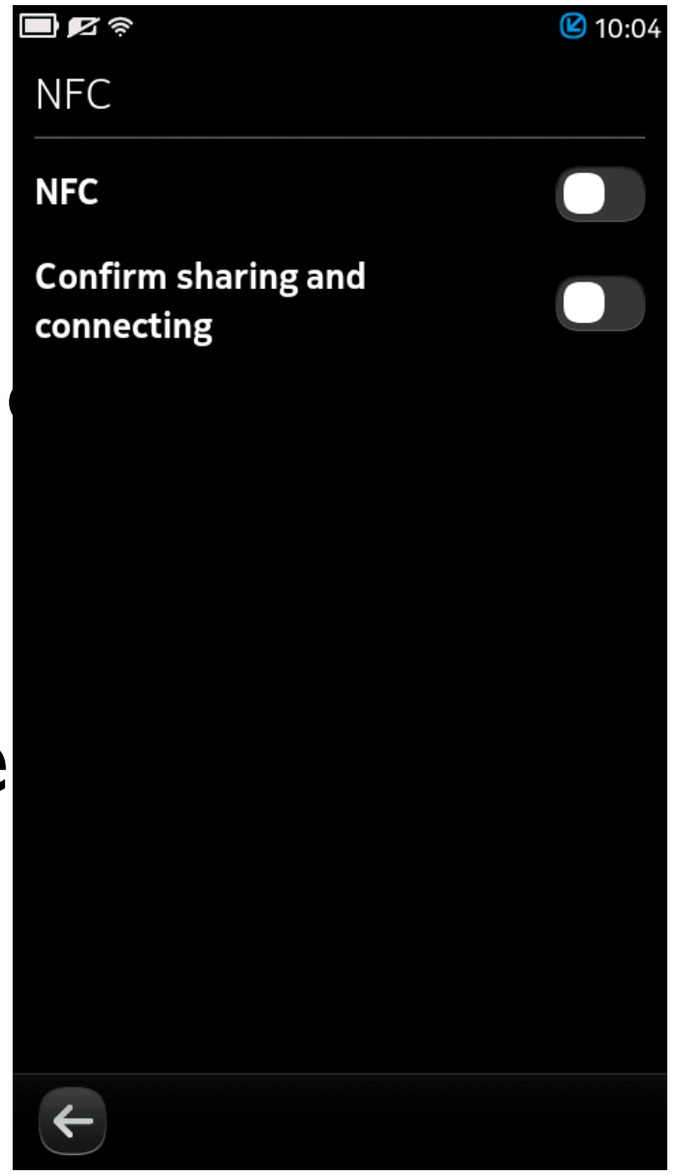
Thanks to Josh Drake and Georg Wicherski (along with entire Crowdstrike team)

Android NFC attack surface



Nokia Content Sharing

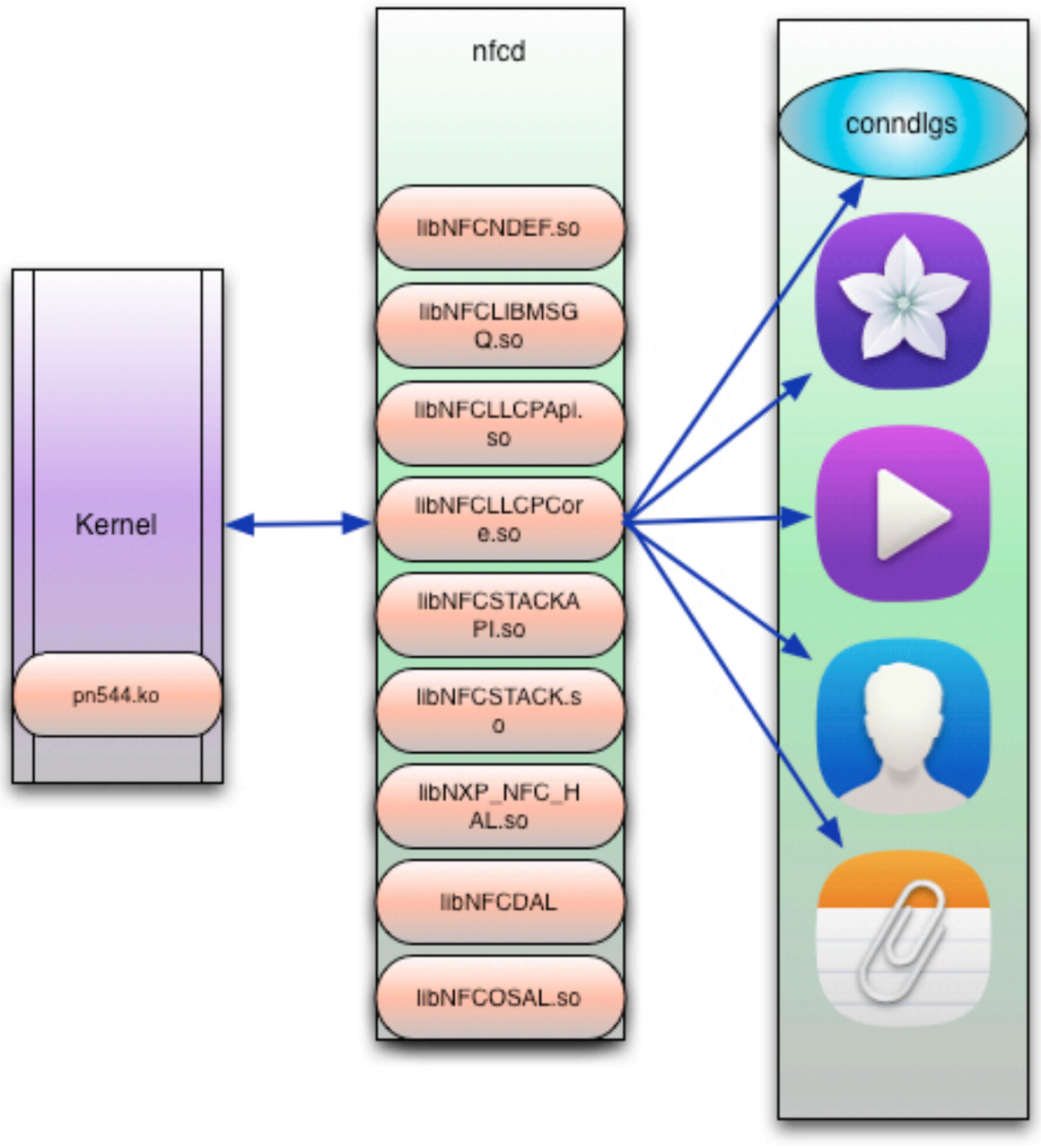
- Like Android Beam for Nokia ph
- Again without user interaction
  - despite what settings would te



Nokia N9 attack surface

App	File type
Contacts	vCard
Gallery	png
	jpg
	gif
	bmp
	tiff
Videos (video-suite)	mp4
	wmv
	3gp
	mp3
	aac
	flac
	wma
	amr
	wav
	ogg
Documents (office-suite)	pdf
	txt
	doc(x)
	xls(x)
	ppt(x)

MeeGo NFC attack surface



Choose public bugs...

- For example, latest N9 firmware ships with libpng 1.2.42

Vulnerability Warning

All "modern" versions of libpng through 1.5.9, 1.4.10, 1.2.48, and 1.0.58, respectively, fail to correctly handle `malloc()` failure for text chunks (in `png_set_text_2()`), which can lead to memory corruption and the possibility of execution of hostile code. This **serious vulnerability** has been assigned ID CVE-2011-3048 and is fixed in version **1.5.10** (and versions **1.4.11** and **1.0.59**, respectively, on the older branches), released 29 March 2012.

Vulnerability Warning

All versions of libpng from 1.0.6 through 1.5.8, 1.4.8, 1.2.46, and 1.0.56, respectively, fail to correctly validate a heap allocation in `png_decompress_chunk()`, which can lead to a buffer-overrun and the possibility of execution of hostile code on 32-bit systems. This **serious vulnerability** has been assigned ID CVE-2011-3026 and is fixed in version **1.5.9** (and versions **1.4.9** and **1.0.57**, respectively, on the older branches), released 18 February 2012.

Or private bugs...

- PPTs

```
==3572== Thread 2:
==3572== Invalid free() / delete / delete[] / realloc()
==3572==    at 0x48347B4: free (vg_replace_malloc.c:366)
==3572==    by 0x5DE780F: free_mem (in /lib/libc-2.10.1.so)
==3572==    by 0x5DE71F7: __libc_freeres (in /lib/libc-2.10.1.so)
==3572==    by 0x48285B7: _vgnU_freeres (vg_preloaded.c:61)
==3572==    by 0x5DB5AC3: __libc_enable_asynccancel (libc-cancellation.c:66)
==3572==    by 0x6826CAF: ??? (in /lib/libglib-2.0.so.0.2800.4)
==3572==  Address 0x7491f30 is not stack'd, malloc'd or (recently) free'd
```

- PDFs

```
==4002== Invalid write of size 1
==4002==    at 0x7290FB4: SplashXPathScanner::clipAALine(SplashBitmap*, int*,
int*, int) (in /usr/lib/libpoppler.so.13.0.0)
==4002==  Address 0xf8dc5090 is not stack'd, malloc'd or (recently) free'd
```

Another MeeGo (Koffice) bug

```
bool STD::read( U16 baseSize, U16 totalSize, OLEStreamReader* stream, bool preservePos )
...
    grupxLen = totalSize - ( stream->tell() - startOffset );
    grupx = new U8[ grupxLen ];
    int offset = 0;
    for ( U8 i = 0; i < cupx; ++i) {
        U16 cbUPX = stream->readU16();  // size of the next UPX
        stream->seek( -2, G_SEEK_CUR ); // rewind the "lookahead"
        cbUPX += 2;                     // ...and correct the size

        for ( U16 j = 0; j < cbUPX; ++j ) {
            grupx[ offset + j ] = stream->readU8();  // read the whole UPX
        }
...
```

koffice-2.3.3/filters/kword/msword-odf/wv2/src/styles.cpp

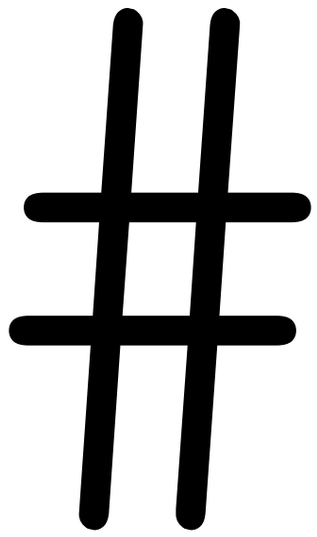
N9 Bluetooth pairing

N9 Bluetooth pairing

- Device will bluetooth pair with another device given a special NDEF message
- Prompts user only if (non-default) “Confirm sharing and connecting” option chosen

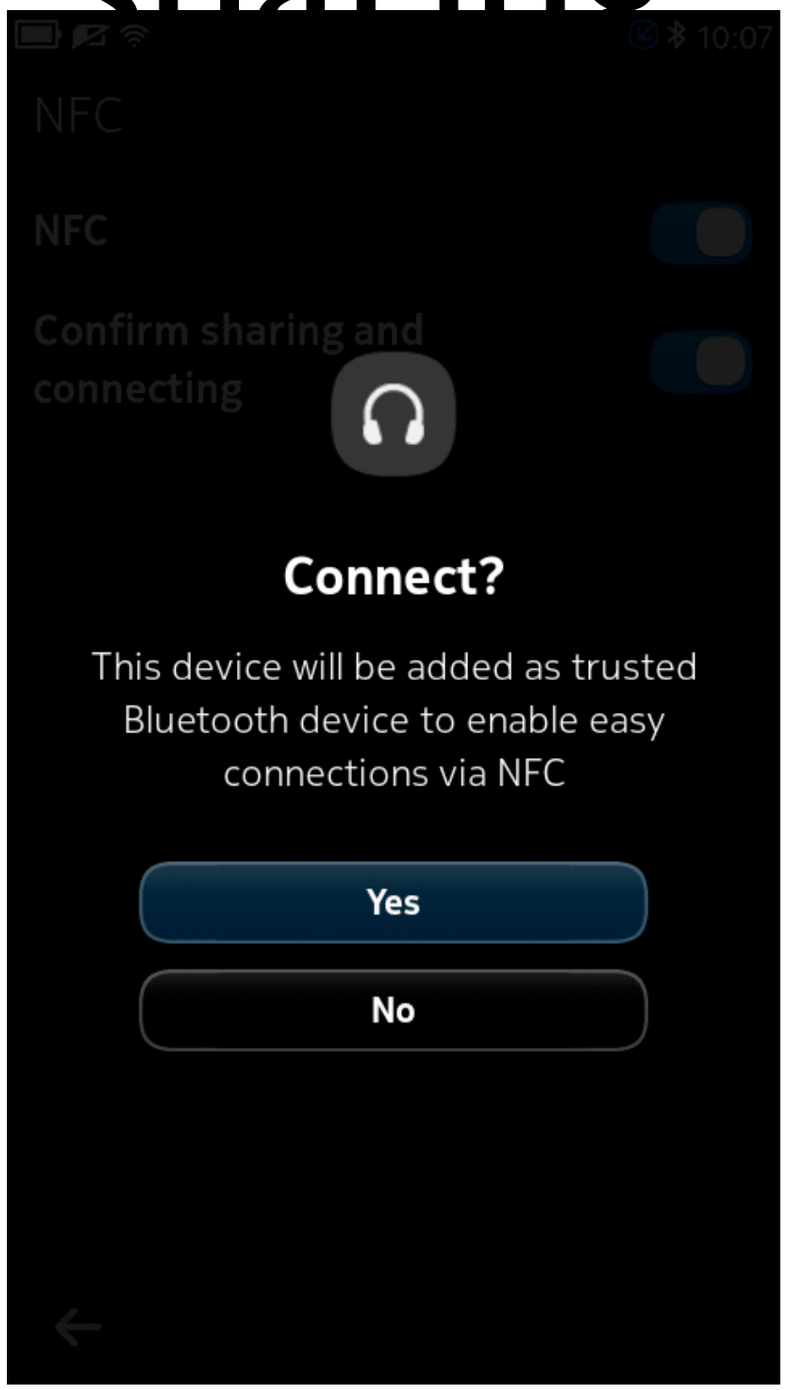
```
[0000] d4 0c 27 6e 6f 6b 69 61 2e 63 6f 6d 3a 62 74 01  ..'nokia.com:bt.
[0010] 00 1d 4f 92 90 e2 20 04 18 31 32 33 34 00 00 00  ..O... ..1234...
[0020] 00 00 00 00 00 00 00 00 00 0c 54 65 73 74 20 6d  ..........Test m
[0030] 61 63 62 6f 6f 6b                                acbook
```

My whole life I've been looking for this



When I should have been looking for this:

Turn on confirm sharing!



Mobile Pwn2Own 2012

The attack

- Samsung phones have S-Beam
- Stock Android only allows web page sharing
- S-Beam allows sharing of many media files
- Researchers from MWR Labs exploited a 0-day flaw in the document viewer

S-Beam

Solutions

- Fix low level bugs
- Turn on user notifications that are VERY specific
  - Prevents subway attack
  - With specific enough notification, prevents skimming attack
  - Hope users are smart enough to defend

Summary

- NFC opens up a new avenue for nearby server-side attacks without user interaction
- NFC stacks are hard to test
  - I released code to help researchers do this
- Vendors should allow option to confirm before NFC data passed to applications

Acknowledgements

- **Accuvant**: Gave me a paycheck while letting me do this work
- **Cyber Fast Track**: Partially funded this research
- **Josh Drake**: Android exploit development
- **Crowdstrike (esp Georg Wicherski)**: For sharing and walking me through their Android browser exploit
- **Michael Ossmann**: GNU Radio help
- **Travis Goodspeed**: Help with N9 basics
- **Kevin Finisterre**: Bluetooth help
- **Corey Benninger and Max Sobell**: GNU Radio and basic NFC stuffs
- **Collin Mulliner**: For trying to help me do NFC memory injection
- **Adam Laurie**: For convincing me I could do card emulation
- **Jon Larimer**: For pointing out one of my crashes corresponded to a fixed double free issue

Questions?

- Contact me
  - cmiller@openrce.org